

Instalando programas no GNU/Linux através do código fonte compilado pelo usuário

Contribuição de Gabriel "Pnordico" Menezes
10 de abril de 2009

Um dos maiores "medos" dos usuários mais novos do sistema operacional GNU/Linux é a instalação manual de programas através do código fonte, disponibilizados através de pacotes tar.bz ou tar.gz. Através desse artigo, tento explicar (e até aprofundar) o processo da compilação de programas e sua instalação, conhecimento este que pode ser necessário, por exemplo, para a instalação de alguns programas que não estejam disponíveis nos repositórios da sua distribuição ou até mesmo para a melhor utilização e entendimento de distribuições como o Slackware e Gentoo.

O primeiro passo para a compilação de um programa no GNU/Linux é o download do código fonte. O código fonte é disponibilizado no site oficial dos projetos normalmente no formato .tar.gz ou .tar.bz2. Uma dica importante é que, dependendo do projeto, existem "tutoriais" no próprio site (ou no wiki do projeto) e dentro do próprio pacote (nos arquivos README e/ou INSTALL) explicando o processo de compilação/instalação do programa.

Para descompactar o arquivo tar.gz:

```
{xtylo_code}$ tar -zxvf <arquivo>.tar.gz{/xtylo_code}
```

Para descompactar o arquivo tar.bz2:

```
{xtylo_code}$ tar -jxvf <arquivo>.tar.bz2{/xtylo_code}
```

Após este comando, um diretório é criado e os arquivos são extraídos para dentro dele, então precisamos acessá-lo:

```
{xtylo_code}$ cd <diretório>{/xtylo_code}
```

A partir desse momento, é importante a leitura do procedimento de instalação do programa, disponível no site ou na wiki do projeto. Caso não exista esse procedimento no site, existe o arquivo README e/ou INSTALL dentro do pacote do programa. Este arquivo de texto dá informações importantes para a compilação e instalação do programa assim como certos argumentos que podem ser passados ao configure (que será explicado mais abaixo) entre outras informações úteis.

Geralmente o procedimento para a compilação e instalação do programa consiste em três comandos:

```
{xtylo_code}$ ./configure --prefix=/usr/local
```

```
$ make
```

```
# make install{/xtylo_code}
```

Chamaremos de "PROCEDIMENTO GERAL", pois se aplica à maioria dos casos.

O próximo passo é seguir o procedimento indicado no README ou no site, o mais comum é que o primeiro comando dado seja o ./configure mas pode variar sendo o ./autogen.sh em alguns programas e em outros isso pode não existir, sendo o primeiro passo o comando make, entre outros, por isso, LEIA O README E/OU O INSTALL.

```
{xtylo_code} $ ./configure {/xtylo_code}
```

É o comando que serve para configurar variáveis, definir configurações dos fontes, verificar a presença dos recursos necessários para a compilação e execução do programa (que tratamos pelo nome de DEPENDÊNCIAS), testar parâmetros e processar os argumentos passados na linha de comando gerando o arquivo Makefile que possui as instruções que serão executadas no comando make.

É no ./configure que você pode passar diversos argumentos para a instalação do programa (argumentos indicados no README e/ou no INSTALL) e as flags de otimização para o compilador.

Para ler uma lista completa desses argumentos utilize o comando "./configure --help"

Argumentos mais comuns:

```
--help
```

```
-h
```

Para mais informações sobre os argumentos existentes e sobre os possíveis argumentos característicos do programa a ser instalado esta é a opção a ser utilizada

--prefix=DIRETÓRIO

Esse argumento é o mais comum, ele é utilizado para definir o diretório onde o programa será "instalado", os diretórios onde os arquivos dos programas serão copiados para que ele possa ser executado. Os subdiretórios da instalação do programa (bin, share, doc, entre outros) são configurados em função da variável definida como PREFIX (ficando sob a forma PREFIX/bin, PREFIX/share, PREFIX/doc). Se você não conhece ou não sabe a utilidade disso, descubra como essa informação pode ser valiosa para garantir a organização do sistema, pois assim os pacotes instalados pelo sistema ou pelo gerenciador de pacotes ficarão em /usr e os instalados pelo usuário ficarão em /usr/local, desse modo o usuário tem mais controle sobre os pacotes que ele instalou manualmente.

Ex: a utilização do comando "./configure --prefix=/usr/local" implicará que os subdiretórios onde os arquivos do programa será copiado estarão dentro do diretório /usr/local, ou seja, este argumento define a variável PREFIX com o valor

'/usr/local'

--enable-FUNÇÃO

--disable-FUNÇÃO

O argumento --enable-FUNÇÃO indica que você estará habilitando alguma função do programa a ser compilado, ou seja, adicionando alguma funcionalidade do programa. Já o --disable-FUNÇÃO serve para desabilitar alguma do código a ser compilado, removendo-a do programa que será utilizado em sua máquina. Recomenda-se utilizar esses argumentos com cautela e muita atenção. Para ter uma lista de algumas das funções que podem ser habilitadas ou desabilitadas em um programa, utilize o comando `./configure --help`

--with-PROGRAMA=DIRETÓRIO

Indica o diretório onde está localizado algum programa ou biblioteca necessária para a compilação ser efetuada com sucesso, geralmente não se faz necessária a utilização deste argumento.

Flags de otimização (CFLAGS e CXXFLAGS)

CFLAGS e CXXFLAGS são variáveis de ambiente que são utilizadas para especificar para o GCC (GNU Compiler Collection) quais "otimizações" utilizar quando compilando o código fonte. CFLAGS são para códigos escritos em C, enquanto CXXFLAGS são para códigos escritos em C++. O GNU gcc handbook mantém uma lista completa das opções disponíveis e suas respectivas funções. Lista de CFLAGS/CXXFLAGS seguras para utilizar caso seu processador seja:

- AMD

- Intel

- PowerPC

- Transmeta

- Via

- Outros Processadores Para saber qual o seu processador e comparar com os disponibilizados nas listas acima, verifique as informações da saída do comando:

```
{xtylo_code} $ cat /proc/cpuinfo {xtylo_code}
```

Exemplo de uso do `./configure` com seus parâmetros:

```
{xtylo_code} $ ./configure --prefix=/usr/local --enable-gtk --disable-qt CFLAGS="-O2 -march=i686 -pipe -fomit-frame-pointer" CXXFLAGS="{CFLAGS}" {xtylo_code}
```

Obs: O comando acima é digitado integralmente em apenas uma linha, constituindo apenas um comando.

A interpretação da saída do `./configure` é fundamental para o sucesso da compilação e instalação do programa. É comum que você tenha de instalar as dependências de algum programa antes de poder compilá-lo e o `./configure` indica essa dependência em sua saída.

Exemplo:

```
{xtylo_code}configure: error: no acceptable C compiler found in $PATH{xtylo_code}
```

Observa-se que o `./configure` indicou que o sistema não possui um recurso necessário para a compilação do código fonte.

Então baixaremos a dependência e instalaremos. Neste ponto vale a pena observar como é importante LER AS ÚLTIMAS LINHAS DO `./configure`, entender essa(s) linha(s) de erro e descobrir o que precisa ser instalado no seu sistema para permitir a compilação.

Uma maneira simples de saber qual pacote deve ser instalado para solucionar o problema é copiar a linha toda, colar no Google e ler os melhores resultados ou então procurar ajuda em fóruns, comunidades ou IRC.

Após um `configure` sem falhas podemos mandar que o programa seja compilado com o comando:

```
{xtylo_code} $ make {xtylo_code}
```

Serão repassadas muitas mensagens de saída. Você não precisa se preocupar enquanto elas passam. Descanse um pouco, procure algo para fazer enquanto o programa é compilado. Quando a compilação concluir o terminal será liberado. Verifique se, dentre as últimas mensagens do `make`, existe alguma de erro. Caso não haja nenhum erro, você pode já executar o programa com um `./<nomedoprograma>` ou partir para a instalação do programa.

Já se houver algum erro você deve parar para analisar a parte final das mensagens que foram geradas durante a compilação, após o comando `make`. Subindo um pouco a barra de rolagem geralmente observa-se que, durante a compilação, o sistema acusou a falta de algum arquivo ou a inexistência de algum comando que não foi detectada no `configure`. Para solucionar este erro é necessário saber o programa que deve ser instalado para que o arquivo ou o comando sejam criados. Para obter essa informação basta copiar e colar o nome do arquivo ou do comando no Google e pesquisar um pouco. Geralmente não é muito difícil encontrar pessoas que já tiveram problema semelhante, mesmo que na compilação de outros programas. Erro no `make` normalmente são bem mais complicados de resolver que problemas no `./configure`, apesar de serem mais raros.

A instalação do programa consiste na cópia dos arquivos compilados para os diretórios onde eles possam ser executados pelos usuários (ou onde for orientado que isso seja feito). O comando para a execução dessa tarefa é o:

```
{xtypo_code} # make install {/xtypo_code}
```

(Observe que o comando é dado como root)

O comando make possui diversos argumentos/funcionalidades:

```
$ make clean
```

Para apagar o que já foi compilado para que possa recomeçar (alguns programas exigem que isso seja feito caso exista algum erro no make)

```
# make uninstall
```

Ao contrário do make install, o make uninstall serve para remover os arquivos já instalados, ou seja, desinstalar o programa.